

parrot Data Science

7th Session – Neural Network(ANN, CNN)

Contents



- Introduction
- What is Neural Network?
- Tensor
- Convolutional Neural Network(CNN)



Introduction

이번 세션에 대한 간략한 소개

Introduction. Schedule

↑ ↓ 2021년 4월

오늘 | 일 | 주 | 월 | 연도별 ...

일요일	월요일	화요일	수요일	목요일	금요일	토요일
03-28	29일 ☀ 13°	30일 ☀	31일 ☀ CNN(1)	04-01 ☀	2일 ☀ 부활절	3일
4일	5일 식목일	6일	7일 CNN(2)	8일	9일 MNIST project	10일

MNIST mini project

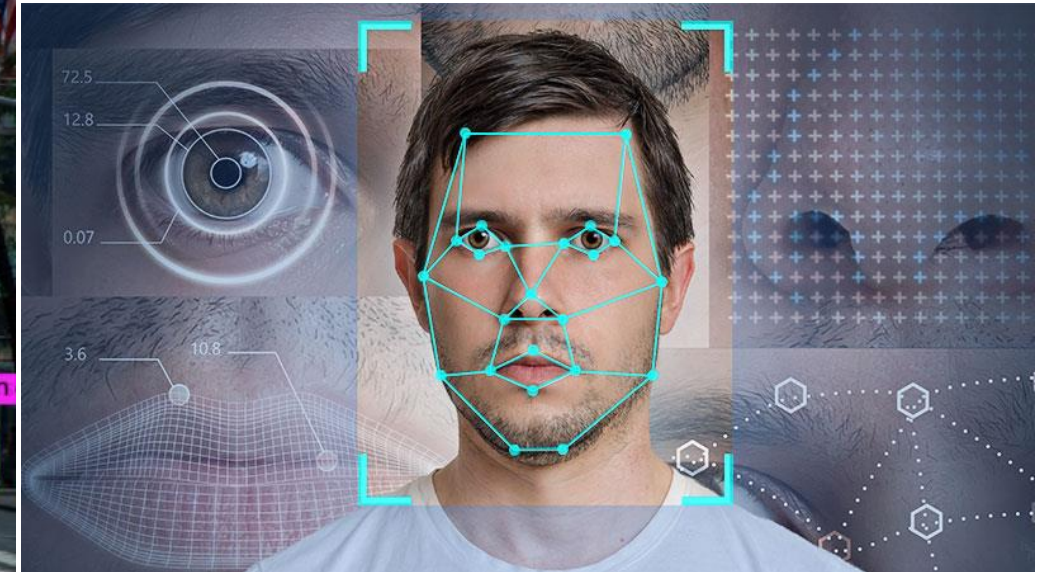
4/9 ~



Simpson image classification
project

5월 예정

Introduction. Computer Vision



어떤 영상에서 장면이나 특징들을
"이해 (Understanding)" 하는
컴퓨터를 프로그래밍하는 것이 목적

- 영상에서 물체의 detection, segmentation
- 같은 장면이나 물체에 대한 다른 관점 (view) 의 등록 (registration)
- 연속 영상에서 물체를 추적
- 어떤 장면을 3차원 모델로 mapping
- 인간의 자세와 팔다리 움직임을 3차원으로 추정 (estimation)

Introduction. Computer Vision

컴퓨터가 사람의 시각 시스템이 하는 작업을 동일하게 수행할 수 있을까?

대용량의 데이터: 연산 속도와 메모리 사용량의 제약
다양한 프로그래밍 스킬로 높은 정확도 확보해야 함



Deep Learning

“여러 층을 가진 인공신경망(Artificial Neural Network, ANN)을
사용해 머신러닝 학습 수행”

→ 층 기반 표현 학습(layered representations learning) 또는 계층적 표현 학습(hierarchical representations learning)



**What is
Neural Network?**

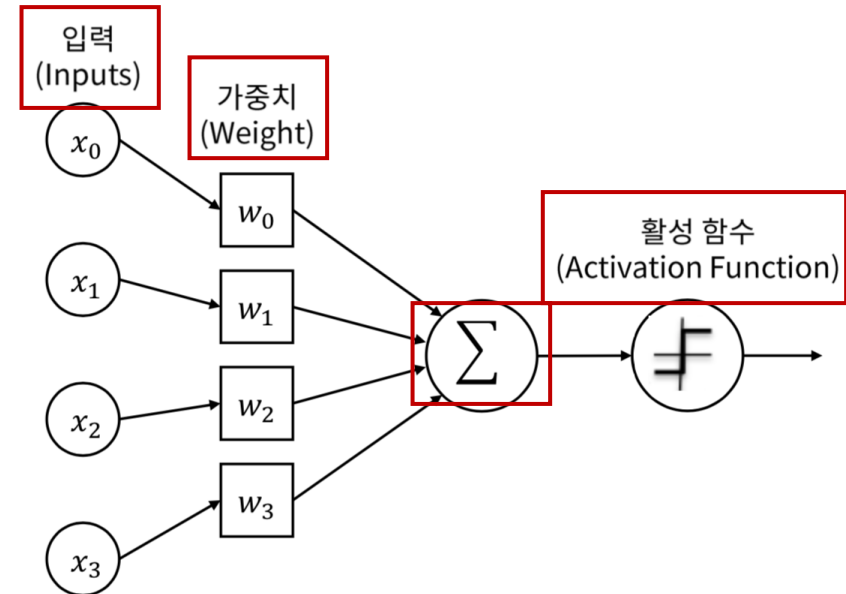
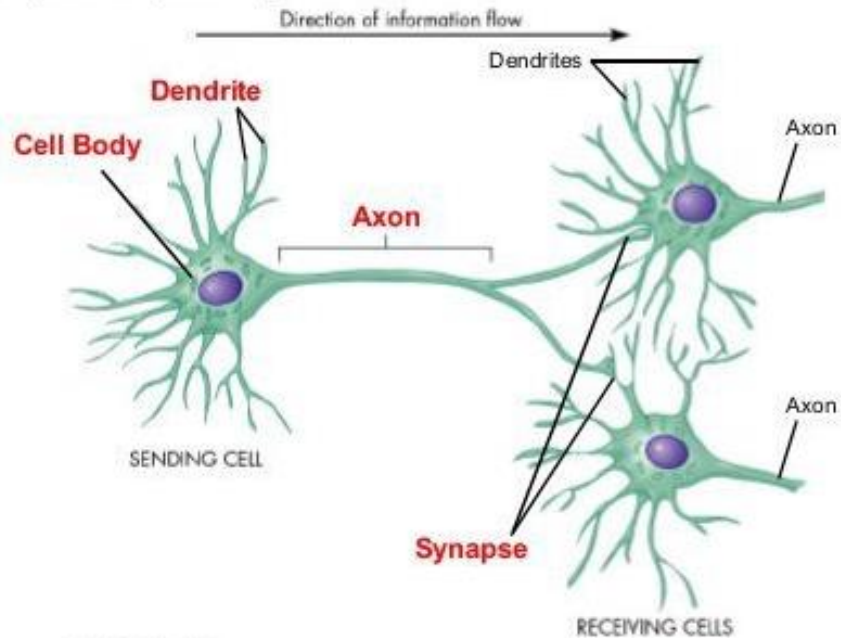
인공신경망에 대한 개괄

Artificial Neural Network

두뇌의 신경세포, 즉 뉴런이 연결된 형태를 모방한 모델

: 인공신경망 뉴런 모델은 생물학적인 뉴런을 수학적으로 모델링한 것

Neuron (Nerve cell) Anatomy



Artificial Neural Network

활성화 함수(Activation function)?

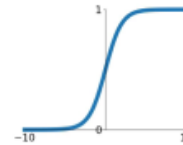
입력 신호의 총합을 출력 신호로 변환하는 함수로,
입력 받은 신호를 얼마나 출력할지 결정하고
네트워크에 층을 쌓아 비선형성을 표현할 수 있도록 해준다.

* 딥러닝 네트워크에서는 노드에 들어오는 값들에 대해 곧바로 다음 layer로 전달하지 않고 주로 비선형인 활성화 함수를 통과시킨 후 전달한다
왜 비선형? → 선형함수를 사용할 시 층을 깊게 하는 의미가 줄어들기 때문

Activation Functions

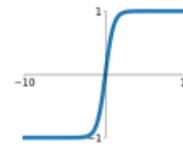
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



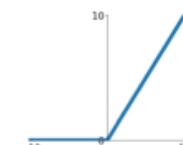
tanh

$$\tanh(x)$$



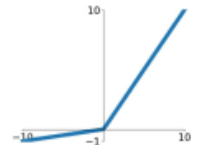
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

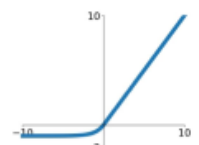


Maxout

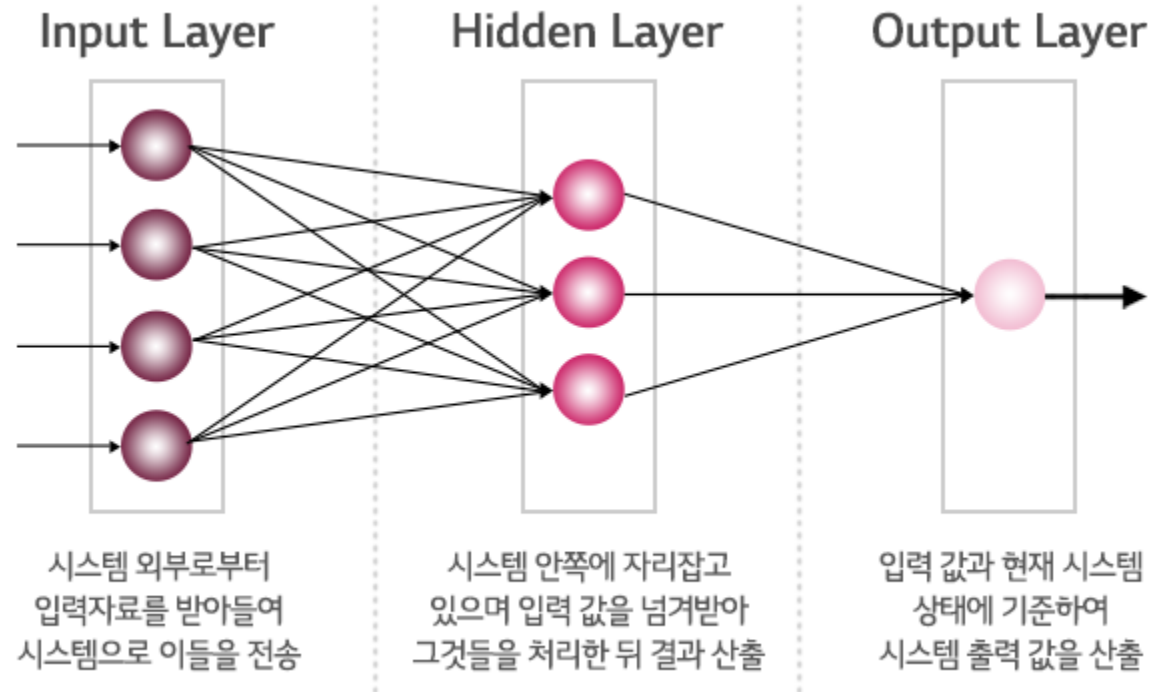
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Artificial Neural Network



- 순전파 *Forwardpropagation*

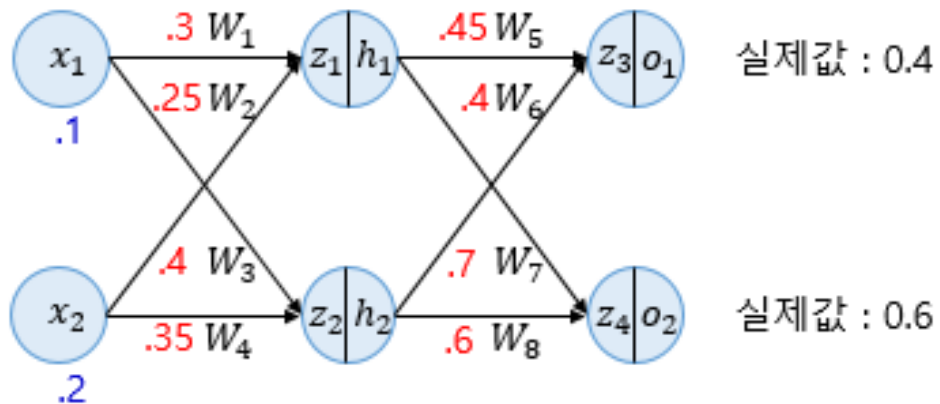
입력층 → 은닉층 → 출력층

- 역전파 *Backpropagation*

출력층 → 은닉층 → 입력층 순의 가중치 업데이트(경사하강법 이용)

Artificial Neural Network

(1) 순전파



- 바이어스 생략
- 2개의 입력층 뉴런, 2개의 은닉층 뉴런, 2개의 출력층 뉴런
- 활성화 함수 : sigmoid
- 학습률 알파는 임의의 수인 0.5로 설정

$$s(z) = \frac{1}{1 + e^{-z}}$$

은닉층의 순입력 함수

$$z_1 = W_1 x_1 + W_2 x_2 = 0.3 \times 0.1 + 0.25 \times 0.2 = 0.08$$

$$z_2 = W_3 x_1 + W_4 x_2 = 0.4 \times 0.1 + 0.35 \times 0.2 = 0.11$$

은닉층의 뉴런 (sigmoid 지남)

$$h_1 = \text{sigmoid}(z_1) = \text{sigmoid}(0.08) = 0.51998$$

$$h_2 = \text{sigmoid}(z_2) = \text{sigmoid}(0.11) = 0.52747$$

출력층의 순입력 함수

$$z_3 = W_5 h_1 + W_6 h_2 = 0.45 \times 0.51998 + 0.4 \times 0.52747 = 0.44498$$

$$z_4 = W_7 h_1 + W_8 h_2 = 0.7 \times 0.51998 + 0.6 \times 0.52747 = 0.68047$$

출력층의 뉴런 (여기도 sigmoid 함수 지남) → 최종 계산, 예측값

$$o_1 = \text{sigmoid}(z_3) = 0.60944$$

$$o_2 = \text{sigmoid}(z_4) = 0.66384$$

* 예측값과 실제값의 오차 (순실함수 - 오차제곱합 MSE)

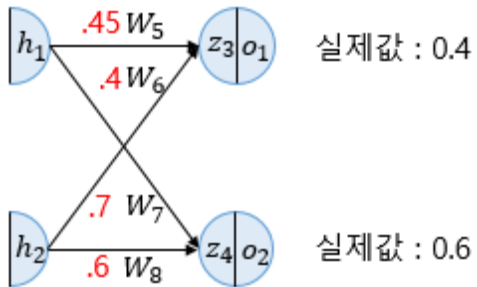
$$E_{o1} = \frac{1}{2} (y_1 - o_1)^2 = 0.02193$$

$$E_{o2} = \frac{1}{2} (y_2 - o_2)^2 = 0.00204$$

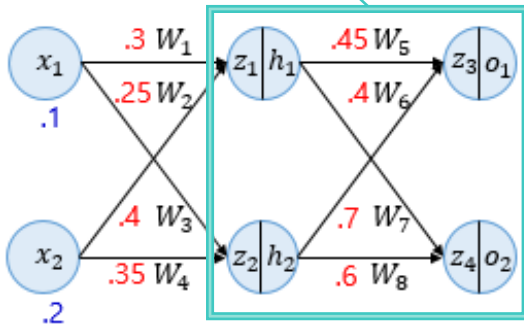
$$E_{total} = E_{o1} + E_{o2} = 0.02397$$

Artificial Neural Network

(1) 역전파 1단계



맨 마지막 층부터!



가중치 W_5 업데이트 하기 $\frac{\partial E_{total}}{\partial W_5}$ 이분의 연쇄법칙 $f(g(x))' = f'(g(x)) \times g'(x)$

$$\frac{\partial E_{total}}{\partial W_5} = \frac{\partial E_{total}}{\partial O_1} \times \frac{\partial O_1}{\partial z_3} \times \frac{\partial z_3}{\partial W_5}$$

① ② ③

① $E_{total} = \frac{1}{2}(y_1 - O_1)^2 + \frac{1}{2}(y_2 - O_2)^2$ 이었으니까

$$\frac{\partial E_{total}}{\partial O_1} = O_1 - y_1 = 0.20944$$

② sigmoid 함수의 미분된 식 $\rightarrow f(x)(1-f(x))$ 나중에 직접 해보기... 위키가 알려줌

$$\frac{\partial O_1}{\partial z_3} = O_1(1-O_1) = 0.60944(1-0.60944) = 0.23802$$

③ $\frac{\partial z_3}{\partial W_5} \rightarrow$ 이거는 위에 순전파 계산할때 z_3 구하는 식으로 보면 그냥 h_1 의 값을 알 수 있음

$$\therefore \frac{\partial E_{total}}{\partial W_5} = 0.20944 \times 0.23802 \times 0.51998 = 0.02592$$

경사하강법을 통해 가중치 업데이트

$$\rightarrow new_W_j = W_j - \alpha \frac{\partial E_{total}}{\partial W_j}$$

이 식은 유로는 제 2번 노트 확인해주세요

$$new_W_5 = W_5 - \alpha \frac{\partial E_{total}}{\partial W_5} = 0.45 - 0.5 \times 0.02592 = 0.43703$$

이런식으로 W_6, W_7, W_8 도 각각 해주면 된다!

$$\frac{\partial E_{total}}{\partial W_6} = \frac{\partial E_{total}}{\partial O_1} \times \frac{\partial O_1}{\partial z_3} \times \frac{\partial z_3}{\partial W_6} \rightarrow new_W_6 = 0.38685$$

$$\frac{\partial E_{total}}{\partial W_7} = \frac{\partial E_{total}}{\partial O_2} \times \frac{\partial O_2}{\partial z_4} \times \frac{\partial z_4}{\partial W_7} \rightarrow new_W_7 = 0.69629$$

$$\frac{\partial E_{total}}{\partial W_8} = \frac{\partial E_{total}}{\partial O_2} \times \frac{\partial O_2}{\partial z_4} \times \frac{\partial z_4}{\partial W_8} \rightarrow new_W_8 = 0.59624$$

Artificial Neural Network

(1) 역전파 2단계



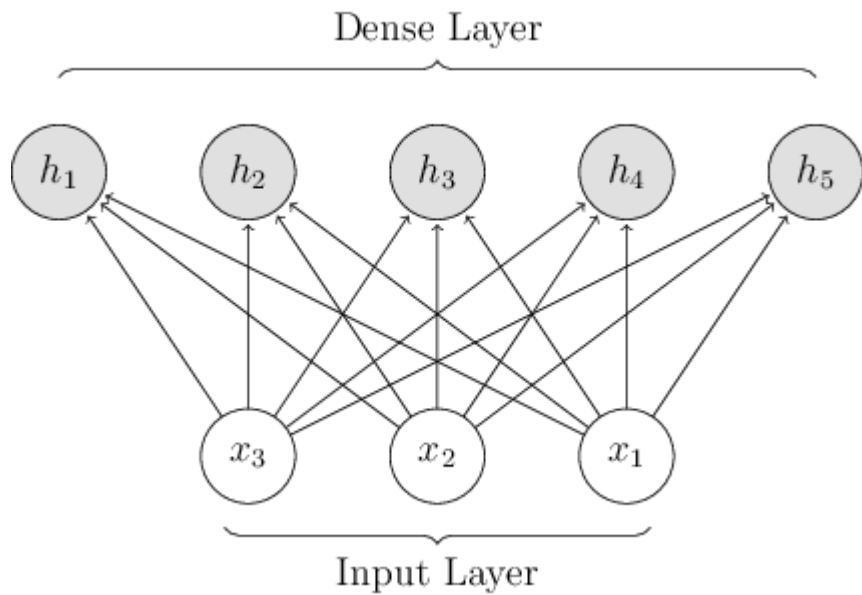
- 위에서 했던 것과 똑같은 방법으로 $W1, W2, W3, W4$ 를 업데이트 해주면 됨
- 모든 가중치에 대한 업데이트가 끝났으면 다시 순전파 방향으로 $o1, o2$ 를 계산해 실제값과 예측값의 오차제곱합을 구해 총 오차합 E_{total} 을 구해준다

newE_total은 기존의 E_total보다 작음
→ 다시 또 순전파와 역전파를 통해 가중치를 업데이트

이렇게 순전파와 역전파를 반복함으로써
오차를 최소화하는 가중치를 찾아간다!!

Artificial Neural Network

Ex) 완전 연결 층 (Fully connected layer)



위 경우는 입력 뉴런이 3개, 출력 뉴런이 5개
= 연결선은 $3 \times 5 = 15$ 개가 됨
각 연결선은 가중치를 포함 - 가중치가 높을수록 해당 입력 뉴런이 출력 뉴런에 미치는 영향이 큼

“16개의 은닉 유닛을 가진 2개의 완전 연결층과, 스칼라 값의 예측을 출력하는 세 번째 층을 쌓아보자”

```
from keras import models
from keras import layers
```

```
model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(1000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

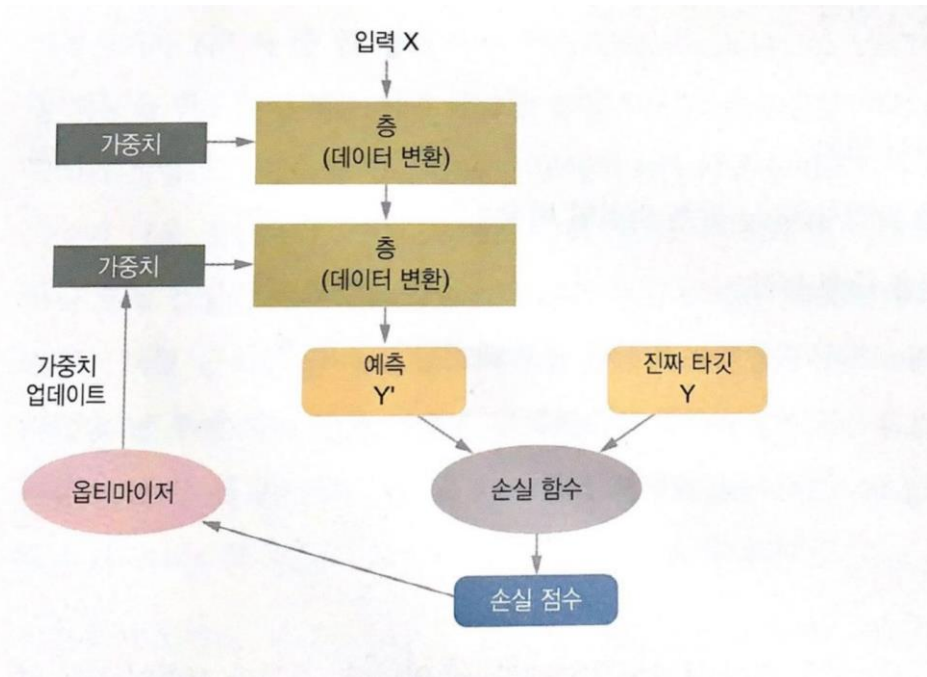
하나의 층

Dense층에 전달한 매개변수(16)는 은닉 유닛의 개수
→ 가중치 행렬 W 의 크기가 (input_dimension, 16)이라는 뜻
→ 입력 데이터와 W 를 점곱하면 입력 데이터가 16차원으로 표현된 공간으로 투영
즉, 은닉 유닛을 늘린다
= 표현 공간을 더 고차원으로 만든다
= 신경망이 더 복잡한 표현을 학습할 수 있다
But, 계산 비용이 커지고 원하지 않는 패턴을 학습함(과적합 주의)

Artificial Neural Network

신경망 훈련에 관련되어 있는 요소들

- 네트워크(또는 모델)를 구성하는 층
- 입력 데이터와 그에 상응하는 타깃
- 학습에 사용할 피드백 신호를 정의하는 손실 함수
- 학습 진행 방식을 결정하는 옵티마이저



Q. 손실 함수(Loss function)?

A. 예측과 타깃을 비교하여 네트워크의 예측이 기댓값에 얼마나 잘 맞는지를 측정하는 손실 값을 만든다. 즉, 훈련하는 동안 최소화될 값!

Q. 옵티마이저(Optimizer)?

A. 손실 함수를 기반으로 네트워크가 어떻게 업데이트될지를 결정한다. 손실 값을 사용하여 네트워크의 가중치를 업데이트!



Tensor

Input 값을 어떻게 매핑할 것인가

Tensor 신경망을 위한 데이터 표현

```
In [3]: import numpy as np
```

```
x = np.array(12)
x
```

```
Out[3]: array(12)
```

```
In [4]: y = np.array([1, 2, 3, 4, 5])
```

```
y
```

```
Out[4]: array([1, 2, 3, 4, 5])
```

```
In [6]: z = np.array([[1, 2, 3, 4, 5],
                     [6, 7, 8, 9, 10],
                     [11, 12, 13, 14, 15]])
z
```

```
Out[6]: array([[ 1,  2,  3,  4,  5],
               [ 6,  7,  8,  9, 10],
               [11, 12, 13, 14, 15]])
```

```
In [8]: p = np.array([[[1, 2, 3, 4, 5],
                       [6, 7, 8, 9, 10],
                       [11, 12, 13, 14, 15]],
                      [[2, 4, 6, 8, 10],
                       [12, 14, 16, 18, 20],
                       [22, 24, 26, 28, 30]],
                      [[1, 3, 5, 6, 9],
                       [11, 13, 15, 17, 19],
                       [21, 23, 25, 27, 29]]])
p
```

```
Out[8]: array([[[ 1,  2,  3,  4,  5],
                 [ 6,  7,  8,  9, 10],
                 [11, 12, 13, 14, 15]],

                [[ 2,  4,  6,  8, 10],
                 [12, 14, 16, 18, 20],
                 [22, 24, 26, 28, 30]],

                [[ 1,  3,  5,  6,  9],
                 [11, 13, 15, 17, 19],
                 [21, 23, 25, 27, 29]])
```

데이터를 위한 수치형 컨테이너(container)

스칼라(0D 텐서)

하나의 숫자만 담고 있는 텐서(0차원)

→ float32, float64 타입의 숫자 *array scalar*

벡터(1D 텐서)

숫자의 배열

행렬(2D 텐서)

벡터의 배열

3D 텐서와 고차원 텐서

행렬들을 배열로 만들면 직육면체 형태로 해석할 수 있는 3D 텐서가 만들어짐

Tensor 신경망을 위한 데이터 표현

Image는
(samples, height, width, channels)
크기의 4D 텐서이다.

이미지는 픽셀 *pixel*로 이루어져 있음.

→ 컬러 사진은 천연색을 표현하기 위해서
각 픽셀을 RGB 3개의 실수로 표현한 데이터

- 축의 개수(랭크): 3D 텐서에는 3개의 축이 있고, 2D 텐서에는 2개가 있다.
- 크기 *shape*: 텐서의 각 축을 따라 얼마나 많은 차원이 있는지를 나타낸 python tuple
- 데이터 타입: dtype(혹은 dtypes)로 확인할 수 있음. float32, uint8, float64 등등

RED Channel



Green Channel



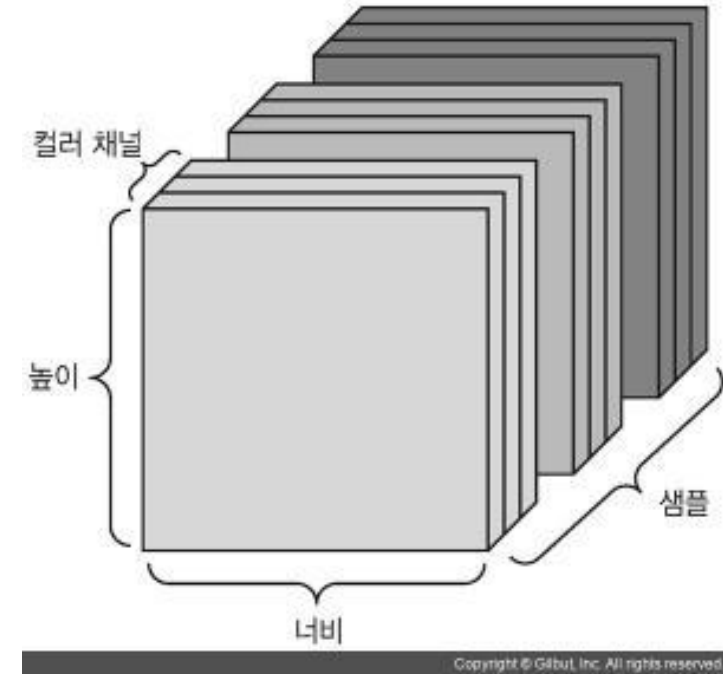
Blue Channel



이미지 출처: [https://en.wikipedia.org/wiki/Channel_\(digital_image\)](https://en.wikipedia.org/wiki/Channel_(digital_image))

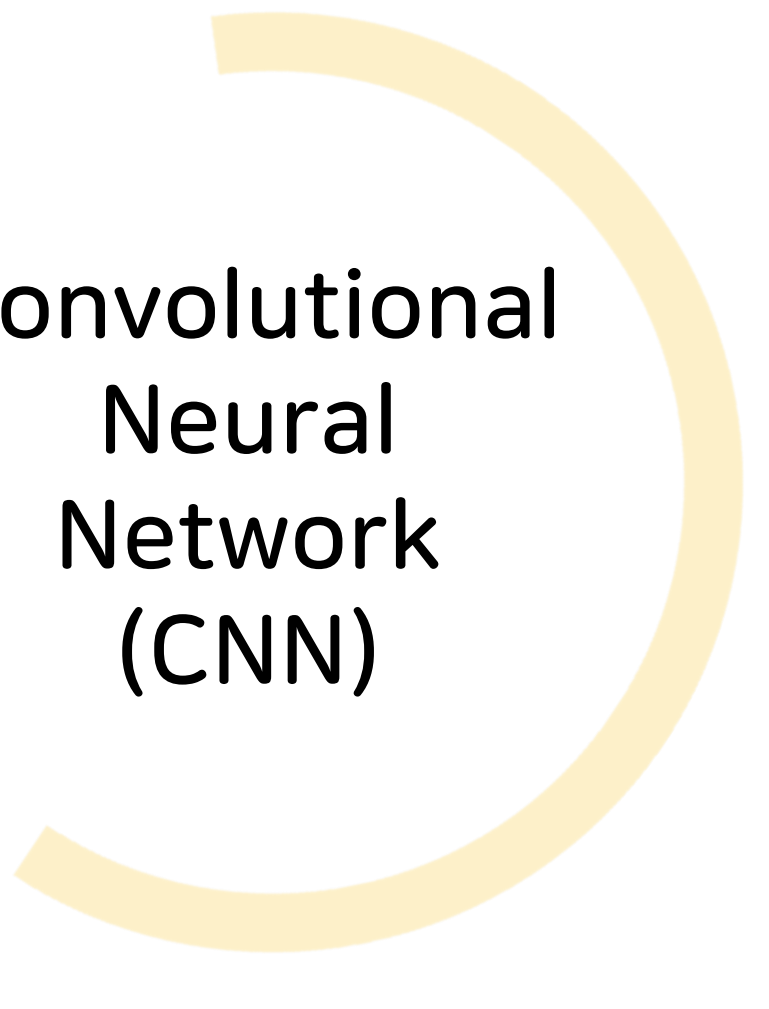
Tensor 신경망을 위한 데이터 표현

데이터 타입	속성
벡터 데이터	(samples, features) 크기의 2D 텐서
시계열 데이터 (Sequence data)	(samples, timestamps, features) 크기의 3D 텐서
이미지	(samples, height, width, channels) 크기의 4D 텐서
동영상	(samples, frames, height, width, channels) 크기의 5D 텐서



텐서의 변환

tensor reshaping → 데이터를 preprocessing 할 때 사용함
: 특정 크기에 맞게 행과 열을 재배치



Convolutional Neural Network (CNN)

합성곱 신경망의 소개

Convolutional Neural Network

층/layer마다 적절한 텐서 포맷과 데이터 처리 방식이 다르다!

- 완전 연결 층(fully connected layer)
- 밀집 연결 층(densely connected layer)
- 순환층(recurrent layer) - LSTM, RNN
- ...

합성곱 층(convolutional layer)

완전 연결 층이나 밀집 연결 층은 1차원 데이터의 처리에만 한정

이미지를 1차원으로 변형을 시키면 데이터의 형상이 소실되는 단점이 있어 이미지 인식에 CNN을 사용하게 되었음

→ 이미지의 텐서는 4D이기 때문에 2D로 변형시키면 정확한 예측을 하기 어려움!

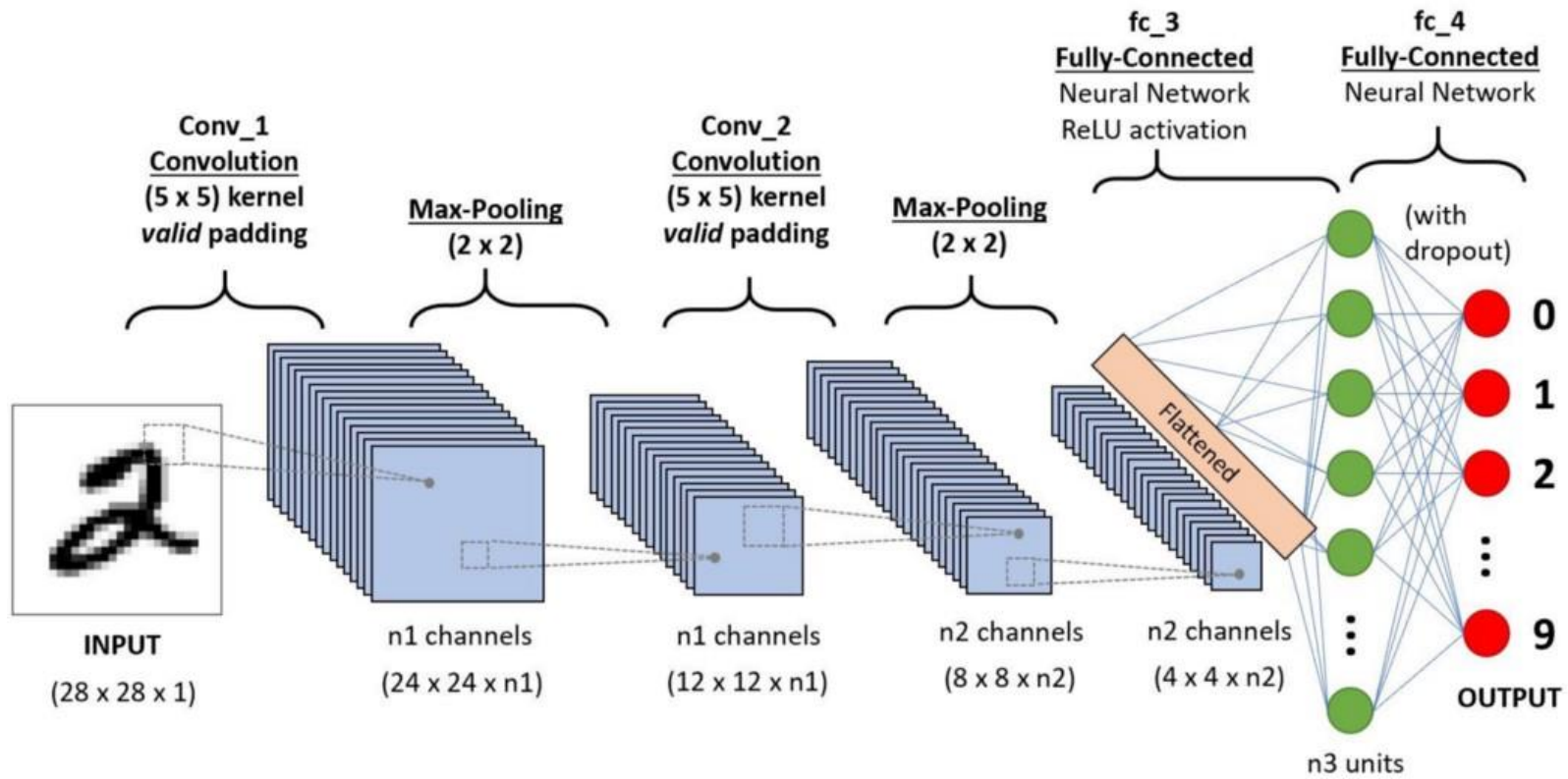
→ 합성곱을 사용하면 3차원 데이터의 공간적인 정보를 유지한 채 다음 레이어로 보낼 수 있음

Convolutional Neural Network

tensorflow.keras.layers에 속해있는 Conv2D 클래스로 구현

```
1 model = tf.keras.models.Sequential()  
2  
3 model.add(tf.keras.layers.experimental.preprocessing.Rescaling(1./255, input_shape=(img_height, img_width, 3)),)  
4 model.add(tf.keras.layers.Conv2D(32, (3, 3), input_shape=(28,28,1),padding='same', activation = "relu"))  
5 model.add(tf.keras.layers.BatchNormalization())  
6 model.add(tf.keras.layers.Conv2D(32, (3, 3), input_shape=(28,28,1),padding='same', activation = "relu"))  
7 model.add(tf.keras.layers.BatchNormalization())  
8  
9 model.add(tf.keras.layers.MaxPool2D())  
10 model.add(tf.keras.layers.Dropout(0.4))  
11  
12 model.add(tf.keras.layers.Conv2D(64, (3, 3), input_shape=(28,28,1),padding='same', activation = "relu"))  
13 model.add(tf.keras.layers.BatchNormalization())  
14 model.add(tf.keras.layers.Conv2D(64, (3, 3), input_shape=(28,28,1),padding='same', activation = "relu"))  
15 model.add(tf.keras.layers.BatchNormalization())  
16  
17 model.add(tf.keras.layers.MaxPool2D())  
18 model.add(tf.keras.layers.Dropout(0.4))  
19  
20 model.add(tf.keras.layers.Flatten())  
21 model.add(tf.keras.layers.Dense(128, activation='relu'))  
22 model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))  
23  
24 model.compile(loss='binary_crossentropy', optimizer='adam',metrics=['accuracy'])  
25
```

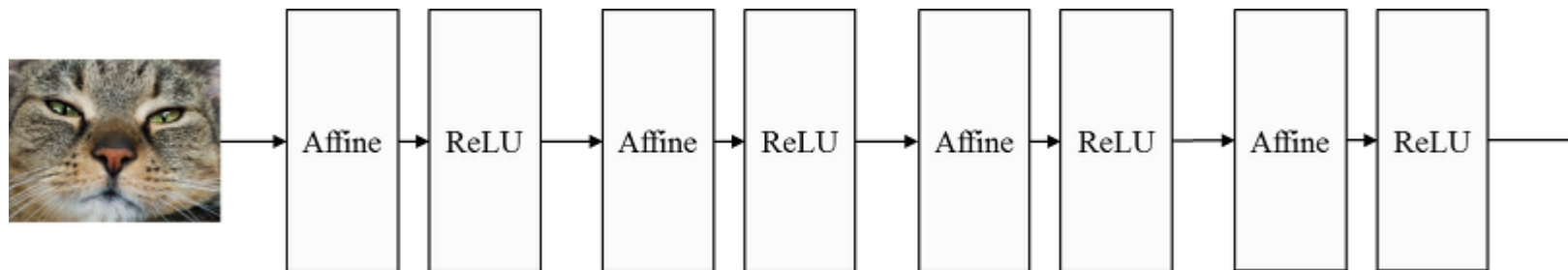
Convolutional Neural Network



합성곱 층, 풀링 층, 완전 연결 층 등으로 구성

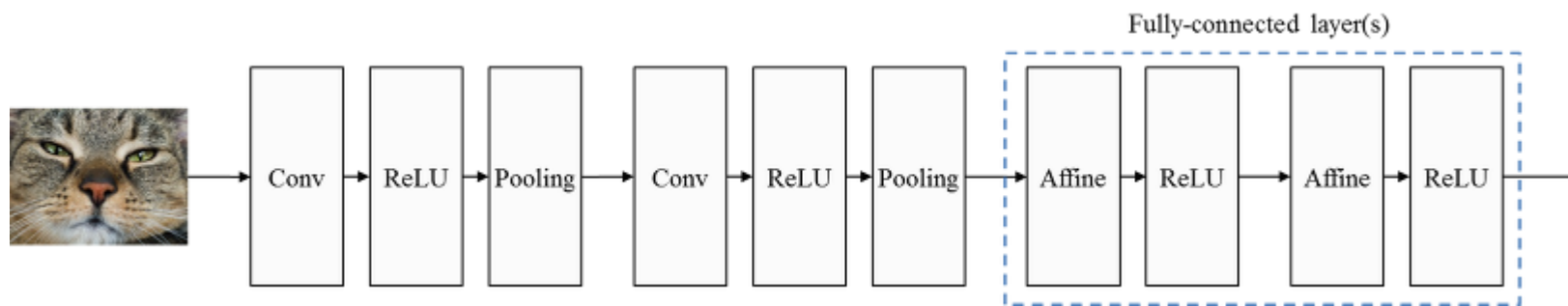
Convolutional Neural Network

완전 연결 신경망



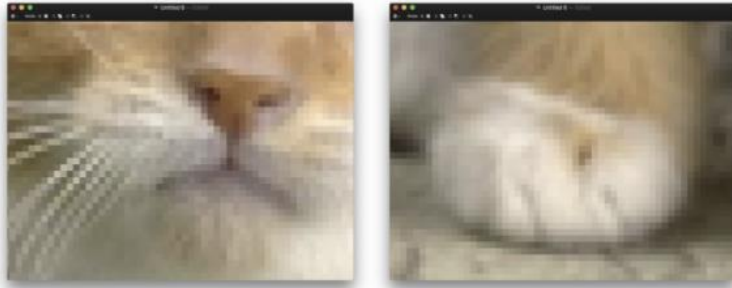
VS

합성곱 신경망



Convolutional Neural Network

점과 선, 질감을 충분히 배우고, 조금 떨어져서 보자.



점과 선이 질감이 합쳐져 삼각형, 동그라미, 복실함이 보인다.

출처: 하용호님 자료 @kakao

삼각형, 원, 사각형, 복실함등을 조합해서 보니



뾰족귀와 땡그란눈과 복실한 발을 배웠다.

출처: 하용호님 자료 @kakao

핵심 특징:

합성곱 층은 지역 패턴을 학습한다.

이미지일 경우 작은 2D window로 입력에서 패턴을 찾음

Ex) 이미지는 에지(edge), 질감(texture) 등의 지역 패턴으로 분해됨

<공간적 계층 구조를 학습>

1st - edge 같은 작은 지역 패턴을 학습

2nd - 첫 번째 층의 특성으로 구성된

더 큰 패턴을 학습

...

복잡하고 추상적인 시각적 개념을 효과적으로 학습

Convolutional Neural Network

How?

Conv2D layer의 filter, stride, padding 파라미터
Pooling layer의 추가 (Maxpooling, AveragePooling...)
Dropout과 BatchNormalization layer의 추가



To be continued